CS 259

Key Exchange Protocols

J. Mitchell

Next few lectures

- Today
  - Key exchange protocols and properties
- Thursday
  - Cathy Meadows: GDOI
- Next Tues
  - Contract-signing protocols
- Next Thurs
  - More about contract signing

Talk about protocols for a while before looking at more tools

Key Management

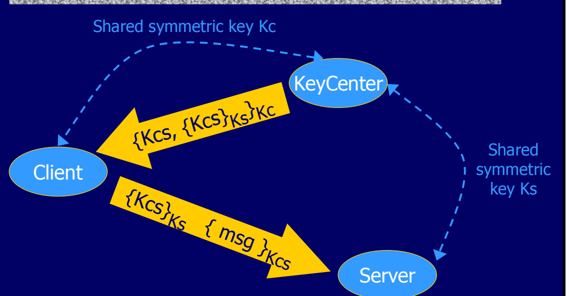
- Out of band
  - Can set up some keys this way (Kerberos)
- Public-key infrastructure (PKI)
  - Leverage small # of public signing keys
- Protocols for session keys
  - Generate short-lived session key
  - Avoid extended use of important secret
  - Don't use same key for encryption and signing
  - Forward secrecy

Cryptography reduces many problems to key management

Internet Standardization Process

- All standards published as RFC (Request for Comment)
  - Available: http://www.ietf.org
  - Not all RFCs are Internet Standards !
- Typical path to standardization
  - Internet Drafts
  - RFC
  - Proposed Standard
  - Draft Standard (requires 2 working implementation)
  - Internet Standard (declared by IAB)
- David Clark, MIT, 1992: "We reject: kings, presidents, and voting. We believe in: rough consensus and running code."

Key Distribution: Kerberos Idea



Key Center generates session key Kcs and distributes using shared long-term keys

Public-Key Infrastructure

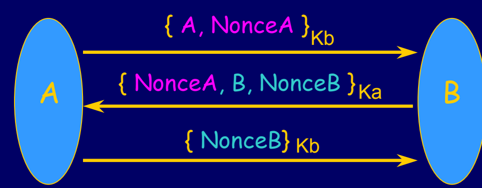
Server certificate can be verified by any client that has CA key Ka

Certificate authority is "off line"

Key Exchange

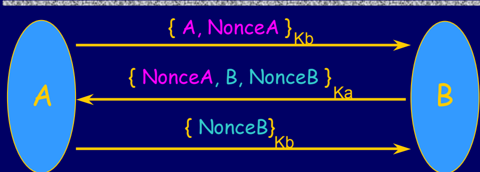
- ◆Parties may have initial information
- ◆Generate and agree on session key
  - Authentication – know ID of other party
  - Secrecy – key not known to any others
  - Avoid replay attack
  - Forward secrecy
  - Avoid denial of service
  - Identity protection – disclosure to others
  - Other properties you can think of???

Needham-Schroeder Lowe



Alice, Bob share two private numbers not known to any observer without Ka^{-1}, Kb^{-1}
Use concatenation (?) or XOR as session key

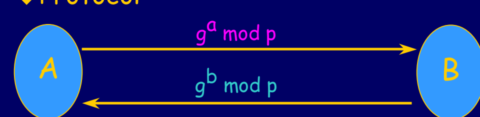
Needham-Schroeder Lowe



Authentication?
Secrecy?
Replay attack
Forward secrecy?
Denial of service?
Identity protection?

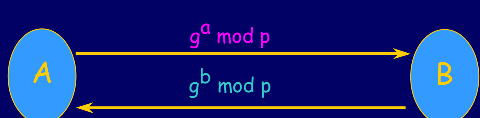
Diffie-Hellman Key Exchange

- ◆Assume finite group $G = \langle S, \bullet \rangle$
  - Generator g so every $x \in S$ is $x = g^n$
  - Example: integers modulo prime p
- ◆Protocol



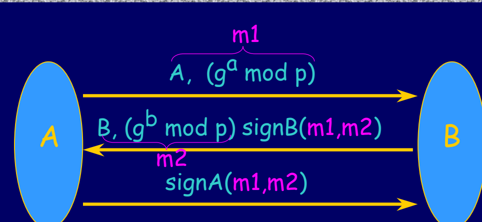
Alice, Bob share g^{ab} mod p not known to anyone else

Diffie-Hellman Key Exchange



Authentication?
Secrecy?
Replay attack
Forward secrecy?
Denial of service?
Identity protection?

IKE subprotocol from IPSEC



Result: A and B share secret g^{ab} mod p

Signatures provide authentication, as long as signature verification keys are known

IPSec: Network Layer Security

- Authentication Header (AH)
  - Access control and authenticate data origins
  - replay protection
  - No confidentiality
- Encapsulated Secure Payload (ESP)
  - Encryption and/or authentication
- Internet Key management (IKE)
  - Determine and distribute secret keys
  - Oakley + ISAKMP
  - Algorithm independent
- Security policy database (SPD)
  - discarded, or bypass

IKE: Many modes

- Main mode
  - Authentication by pre-shared keys
  - Auth with digital signatures
  - Auth with public-key encryption
  - Auth with revised public-key encryption
- Quick mode
  - Compress number of messages
  - Also four authentication options

Aug 2001 Position Statement

- In the several years since the standardization of the IPSEC protocols (ESP, AH, and ISAKMP/IKE), … several security problems…, most notably IKE.
- Formal and semi-formal analyses by Meadows, Schneier et al, and Simpson, have shown … security problems in IKE stem directly from its complexity.
- It seems … only a matter of time before serious *implementation* problems become apparent, again due to the complex nature of the protocol, and the complex implementation that must surely follow.
- The Security Area Directors have asked the IPSEC working group to come up with a replacement for IKE.

How to study complex protocol

General Problem in Security

- Divide-and-conquer is fundamental
  - Decompose system requirements into parts
  - Develop independent software modules
  - Combine modules to produce required system
- Common belief:
  - Security properties do not compose

Difficult system development problem

Example protocol

Protocol P1

$A \rightarrow B : \{message\}_{KB}$

$A \rightarrow B : KA^{-1}$

- This satisfies basic requirements
  - Message is transmitted under encryption
  - Revealing secret key KA^{-1} does not reveal message

Similar protocol

Protocol P2

$B \rightarrow A : \{message'\}_{KA}$

$B \rightarrow A : KB^{-1}$

- Transmits msg securely from B to A
  - Message is transmitted under encryption
  - Revealing secret key KB^{-1} does not reveal message

Composition P1; P2

- Sequential composition of two protocols

$A \rightarrow B : \{message\}_{KB}$

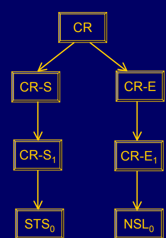
$A \rightarrow B : KA^{-1}$

$B \rightarrow A : \{message'\}_{KA}$

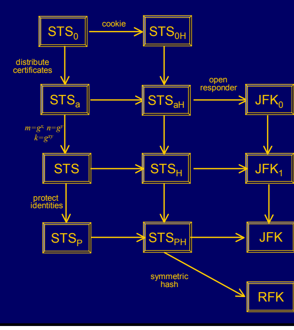
$B \rightarrow B : KB^{-1}$

- Definitely not secure
  - Eavesdropper learns both keys, decrypts messages

Basic challenge-response



STS family



Example

- Construct protocol with properties:
  - Shared secret
  - Authenticated
  - Identity Protection
  - DoS Protection
- Design requirements for IKE, JFK, IKEv2 (IPSec key exchange protocol)

Component 1

- Diffie-Hellman

$A \rightarrow B: \; g^a$

$B \rightarrow A: \; g^b$

  - Shared secret (with someone)
    - A deduces:

$Knows(Y, g^{ab}) \supset (Y = A) \vee Knows(Y,b)$

  - Authenticated
  - Identity Protection
  - DoS Protection

Component 2

- Challenge Response:

  A → B: m, A
  B → A: n, sig_B {m, n, A}
  A → B: sig_A {m, n, B}

  - Shared secret (with someone)
  - Authenticated
    - A deduces: Received (B, msg1) ∧ Sent (B, msg2)
  - Identity Protection
  - DoS Protection

Composition

m := g^a
n := g^b

- ISO 9798-3 protocol:

  A → B: g^a, A
  B → A: g^b, sig_B {g^a, g^b, A}
  A → B: sig_A {g^a, g^b, B}

  - Shared secret: gab
  - Authenticated
  - Identity Protection
  - DoS Protection

Refinement

- Encrypt signatures:

  A → B: g^a, A
  B → A: g^b, E_K {sig_B {g^a, g^b, A}}
  A → B: E_K {sig_A {g^a, g^b, B}}

  - Shared secret: gab
  - Authenticated
  - Identity Protection
  - DoS Protection

Transformation

- Use cookie: JFK core protocol

  A → B: g^a, A
  B → A: g^b, $hash_{KB}$ {g^b, g^a}
  A → B: g^a, g^b, $hash_{KB}$ {g^b, g^a}
  E_K {sig_A {g^a, g^b, B}}
  B → A: g^b, E_K {sig_B {g^a, g^b, A}}

  - Shared secret: gab
  - Authenticated
  - Identity Protection
  - DoS Protection

(Here B must store b in step 2, but we'll fix this later...)

Cookie transformation

- Typical protocol
  - Client sends request to server
  - Server sets up connection, responds
  - Client may complete session or not (DOS)
- Cookie version
  - Client sends request to server
  - Server sends hashed data back
    - Send message #2 later after client confirms
  - Client confirms by returning hashed data
  - Need extra step to send postponed message

Cookie in JFK

- Protocol susceptible to DOS

  A → B: g^a, A
  B → A: g^b, E_K {sig_B {g^a, g^b, A}} (eh1)
  A → B: E_K {sig_A {g^a, g^b, B}} (eh2)

- Use cookie: JFK core protocol

  A → B: g^a, A
  B → A: g^b, $hash_{KB}$ {g^b, g^a}
  A → B: g^a, g^b, $hash_{KB}$ {g^b, g^a}, eh2
  B → A: g^b, eh1

Efficiency: Reuse D-H key

- Costly to compute g^a, g^b, g^{ab}
- Solution
  - Keep medium-term g^a, g^b (change ~10 min)
  - Replace g^a by pair g^a, nonce
- JFKi, JFKr protocols (except cert or grpinfo, …)

  A → B: Na, g^a, A

  B → A: Nb, g^b, $hash_{KB}$ {Nb, Na, g^b, g^a}

  A → B: Na, Nb, g^a, g^b, $hash_{KB}$ {Nb, Na, g^b, g^a},
  E_K {sig_A {Na, Nb, g^a, g^b, B}}

  B → A: g^b, E_K {sig_B {Na, Nb, g^a, g^b, A}}

Note: B does not need to store any short-term data in step 2

Conclusion

- Many protocol properties
  - Authentication Secrecy
  - Prevent replay Forward secrecy
  - Denial of service Identity protection
- Systematic understanding is possible
  - But be careful; easy to make mistakes
  - State of the art:
    need to analyze complete protocol